Siegfried-Angel Gevatter Pujals
Llenguatges de programació - FIB - Primavera 2012

PL/SQL

Breu presentació del llenguatge

En aquest document presentaré breument el llenguatge PL/SQL. Començaré donant una idea de l'estructura bàsica d'un programa PL/SQL, els tipus de variables i les diferents estructures de control. A continuació, mostraré algun exemple de com combinar el llenguatge amb SQL. Finalment, presentaré algunes funcionalitats interessants com ara classes, introspecció i reflexió.

Siegfried-Angel Gevatter Pujals

Introducció

El PL/SQL (Procedural Language/SQL) és un llenguatge de programació procedural desenvolupat per Oracle per ser usat conjuntament amb les seves bases de dades relacionals. Actualment funciona amb les bases de dades Oracle, TimesTen i la DB2 d'IBM.

Es tracta d'un llenguatge imperatiu construit al voltant del llenguatge SQL, amb síntaxi similar a Ada o Pascal. Va ser alliberat l'any 1988 i l'any 1997, amb la versió 8 d'Oracle Database, va guanyar suport per al paradigma de desenvolupament orientat a objectes.

El tipatge i els àmbits són estàtic i, com veurem més endavant, té tipatge dèbil.

PL/SQL s'acustoma a utilitzar de forma interpretada, tot i que en les seves últimes versions també té la possibilitat de ser compilat a codi nadiu. En ambdos casos, l'execució té lloc dins el procés de la base de dades, de manera que l'ús de consultes SQL és molt eficient. De fet, és habitual desar el propi programa PL/SQL (ja sigui en forma de codi o binaria) dins la mateixa base de dades, de manera que es pugui invocar des de crides SQL i que el seu ús quedi protegit per l'estructura d'usuaris i permissos de la base de dades.

Tot i tractar-se d'un llenguatge d'extensió per a bases de dades, es poden arribar a fer bastantes coses en PL/SQL. De fet, és possible utilitzar-lo per desenvolupar pàgines web.

Una primera visió del llenguatge

Estructura bàsica

Començem veient un exemple de programa:

```
DECLARE
      hola              VARCHAR2(30);
      missatge          VARCHAR2(30);
BEGIN
      hola := 'Hola';
      -- Concatenació
      missatge := hola || 'mon!';
      -- Escriptura
      DBMS_OUTPUT.PUT_LINE(missatge);
EXCEPTION
      ...
END;
/
```

En aquest codi podem veure l'estructura bàsica d'un bloc de codi. Tot bloc PL/SQL ha d'estar format per les paraules clau `BEGIN` i `END` al voltant de les instruccions que el formen. Si es vol utilitzar variables, aquestes s'han de declarar abans en una secció addicional amb la paraula clau `DECLARE`. Finalment, es recomana tancar amb una secció `EXCEPTION` que serà l'encarregada de dur a terme el tractament d'errors. És possible imbricar un bloc de codi dins d'un altre.

En aquest exemple declarem dues variables que puguin contenir cadenes de text de fins a 30 caràcters. A la primera li assignem la cadena "Hola" i a la segona la concatenació de la primera variable i el text "mon!".

Finalment, escrivim el resultat utilitzant la funció `DBMS_OUTPUT.PUT_LINE`. Podem executar aquest codi utilitzant l'eina de línia de comandes SQL*Plus. Abans però haurem d'executar l'instrucció `SET SERVEROUTPUT ON` per tal d'indicar-li que ens mostri el buffer de sortida.

A l'executar codi de forma interactiva, és possible obtenir entrades de l'usuari mitjançant l'ús del que anomenen *variables de substitució*. Bàsicament, n'hi ha prou amb prefixar el nom d'una variable amb el signe "*&*" i SQL*Plus mostrarà un dialeg demanant quin valor se li ha d'assignar.

Tipus de dades

Alguns dels tipus de dades bàsics que suporta PL/SQL són els següents:

```
DEC / DECIMAL            CHAR / CHARACTER         BOOLEAN
FLOAT                    VARCHAR2                 DATE
DOUBLE PRECISION         NCHAR                    TIMESTAMP
REAL                     NVARCHAR2
                         STRING                   REF
INTEGER                  LONG                     REF CURSOR
POSITIVE
NATURAL                  RAW                      BFILE
PLS_INTEGER                                       BLOB

NUMBER
```

Com es pot veure, la consistència pel que fa al disseny dels tipus de dades no és el punt fort del PL/SQL.

No entrarem gaire en detalls, però comentem algun d'aquests tipus... `CHAR` és per a cadenes de text de mida constant, `VARCHAR2` és per a cadenes de text de mida variabla (`VARCHAR` no existeix, sinó que està reservat per a ús futur) i les variants `NCHAR` i `NVARCHAR2` són el mateix per a caràcters unicode.

Pel que fa als numerics, `DECIMAL` és per a decimals de punt fixe i els tres tipus que el segueixen són de coma flotant. `PLS_INTEGER` és un número de 32-bits (i utilitza les instruccions aritmètiques nadiues del processador), `NATURAL` i `POSITIVE` restringeixen el valor que poden prendre els números. `NUMBER` és el tipus més general i es pot configurar al ser utilitzat.

El llenguatge té conversió implícita entre tipus de dades de nombres i de text, la qual pot resultar en errors sí el valor no és convertible.

Com a tipus contenidors tenim els `VARRAY` (arrays de mida variable però amb un tamany màxim prefixat), els `RECORD` (tipus de dades estructurats) i tipus (com ara els *nested table* o els *associative arrays*) construits al voltant del concepte de taula SQL.

Estructures de control

Pel que fa a les estructures de control no hi ha cap gran sorpresa:

- IF … ELSIF … ELSE
- CASE … WHEN … ELSE
- FOR … IN … LOOP
- WHILE … LOOP
- LOOP … EXIT WHEN

Funcions i procediments

L'exemple que hem vist abans constituia un bloc anònim. També és possible emmagatzemar blocs de codi a la base de dades, en forma de procediments o funcions.

La forma de definir un procediment és la següent:

```
CREATE [OR REPLACE] PROCEDURE nom(
                  p_var1 IN tipus1,
                  p_var2 OUT tipus2,
                  p_var3 IN OUT tipus3)
IS
      v_var4 tipus4;
BEGIN
      . . .
EXCEPTION
      . . .
END;
```

El prefix dels noms de paràmetre i variable ("p_" i "v_") és simplement una convenció; és possible utilitzar noms qualsevol. Com es pot apreciar, els paràmetres poden ser d'entrada i/o sortida.

Una funció és com un procediment però retorna un valor de sortida (independent dels possibles paràmetres de sortida que pugui tenir). Simplement canvia la sintaxi per la següent: `CREATE [OR REPLACE] FUNCTION nom (...) RETURN tipus_retorn`.

Per utilitzar un procediment (o funció), aquest ha d'estar declarat abans. Si això suposa un problema, és possible emprar pre-declaracions escrivint la capçalera del procediment sense incloure el cos (és a dir: `PROCEDURE nom (...);`).

Interacció amb la base de dades

Consultes SQL

La característica principal del PL/SQL és que permet incorporar consultes SQL de forma molt natural. De fet, es poden executar com una instrucció més. El que les fa realment interessants és que et pot guardar el resultat d'una consulta dins d'una variable:

```
DECLARE
      deutes      NUMBER(9, 2);
BEGIN
      SELECT SUM(invoice_total - payment_total - credit_total)
      INTO deutes
      FROM invoices
      WHERE vendor_id = 95

      IF deutes > 0 THEN
            DMBS_OUTPUT.PUT_LINE('Em deu: $' || ROUND(deutes, 2));
      ELSE
            DBMS_OUTPUT.PUT_LINE('Tot pagat');
      END IF
END;
```

També podem utilitzar variables de forma transparent. Per exemple, si a l'exemple anterior tinguessim una segona variable "`venedor INTEGER`" podriem substituir la condicció de la consulta per "`WHERE vendor_id = venedor`".

Convenientment, si tenim una consulta SQL que es repeteix (per exemple, perquè es troba dins d'un bucle), després d'executar-la per primera vegada Oracle la mantindrà en un *pool* compartit per tal que no calgui tornar a compilar-la de zero les següents vegades.

Disparadors

Si volem que un cert codi s'executi cada cop que passa alguna cosa a la base de dades (s'insereix/modifica/esborra una fila d'una certa taula), podem definir-ho un disparador amb:

```
CREATE OR REPLACE TRIGGER nom_disparador
BEFORE INSERT ON nom_taula
BEGIN
      . . .
END;
/
```

A diferència dels llenguatges d'altres sistemes gestors de bases de dades, PL/SQL també permet disparadors compostos, com ara:

```
CREATE OR REPLACE TRIGGER nom_disparador
FOR INSERT ON nom_taula
COMPOUND TRIGGER
      variable    NUMBER;

      BEFORE STATEMENT IS
      BEGIN
            /* ... */
      END BEFORE STATEMENT;

      AFTER STATEMENT IS
      BEGIN
            /* ...*/
      END AFTER STATEMENT;
END nom_disparador;
/
```

En aquest exemple, la variable declarada al principi permetria compartir dades entre els diferents disparadors imbricats.

Algunes altres funcionalitats

Orientació a objectes

Si volem crear una classe en PL/SQL hem de definir-ne la capçalera i el cos per separat. Per exemple, podem començar definint una capçelera com la següent:

```
CREATE OR REPLACE TYPE preu_obj AS OBJECT (
      preu              NUMBER(10, 2),
      descompte         NUMBER(10, 4),

      CONSTRUCTOR FUNCTION preu_obj (price NUMBER)
            RETURN SELF AS RESULT
            MEMBER FUNCTION preu_amb_descompte RETURN NUMBER)
INSTANTIABLE
FINAL;

CREATE OR REPLACE TYPE producte AS OBJECT (
      id_producte       NUMBER(10),
      preu              PREU_OBJ);
```

I per definir-ne la implementació fariem el següent:

```
CREATE OR REPLACE TYPE BODY preu_obj AS
      CONSTRUCTOR FUNCTION preu_obj (price NUMBER)
            RETURN SELF AS RESULT
            IS
      BEGIN
            SELF.preu := preu;
            RETURN;
      END;

            MEMBER FUNCTION preu_amb_descompte RETURN NUMBER
            IS
            BEGIN
                  ...
            END;
END;
```

Si no volem utilitzar classes però ens agradaria encapsular un conjunt de funcions i procediments, PL/SQL també permet agrupar-les en *paquets*.

Introspecció

És curiós que PL/SQL proporciona funcions per obtenir informació sobre l'execució actual. En particular, dins d'una funció podem veure quina funció l'ha cridat, utilitzant la crida `owa_util.who_called_me` o `dbms_utility.format_call_stack`.

També podem obtenir informació sobre quins procediments i funcions emmagatzemats existeixen. Donat que, com hem comentat, es desen a la base de dades, és possible executar consultes com ara:

```
SELECT * FROM {ALL_OBJECTS,DBA_OBJECTS,USER_OBJECTS}
WHERE owner = 'user2' AND object_type = 'FUNCTION';
```

Reflexió

Més útil que el que hem comentat d'introspecció, és la possibilitat de construir codi (ja sigui SQL simple o PL/SQL) de forma dinàmica mitjançant concatenació de cadenes. Per exemple, si tenim una variable `codi VARCHAR2`, podriem fer el següent:

```
codi := 'CREATE PROCEDURE ' || p_region || '_inserter IS BEGIN … END;';
EXECUTE IMMEDIATE codi;
```

Cal tenir en compte que quan construim consultes de forma dinàmica són vistes com a consultes úniques. Si volem evitar que s'hagi de compilar cada cop, hem d'utilitzar *bind variables*. Veiem aquí un exemple:

```
codi := 'INSERT INTO departments VALUES (:1, :2, :aa, :bb);';
EXECUTE IMMEDIATE codi USING dept_id, 'department X', 10, foo;
```

Bibliografia

- Wikipedia
  http://en.wikipedia.org
- PL/SQL Tutorial
  http://plsql-tutorial.com
- PL/SQL User's Guide and Reference
  http://docs.oracle.com/cd/B13789_01/appdev.101/b10807/toc.htm
- Yu-May Chang & Jeff Ullman, Using Oracle PL/SQL
  http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html
- Lakshman Bulusu: Oracle PL/SQL: Expert Techniques for Developers and Database Administrators
  Charles River Media (2008)
- Joel Murach, Murach's Oracle SQL and PL/SQL
  Mike Murach & Associates (2008)
- Michael McLaughlin, Oracle Database 11g PL/SQL Programming
  McGraw-Hill Osborne Media (2008)
- Benjamin Rosenzweig & Elena Silvestrova, Oracle PL/SQL Interactive Workbook
  Person Education (2000)
- Timothy Hall, Oracle PL/SQL Tuning: Expert Secrets for High Performance Programming
  Rampant TechPress (2006) - http://www.dba-oracle.com/plsql/t_plsql_cursor_variables.htm
- Oracle PL/SQL examples
  http://www.java2s.com/Code/Oracle/CatalogOracle.htm
- The Oracle FAQ
  http://www.orafaq.com/
- Beng Chin, PL/SQL
  http://www.comp.nus.edu.sg/~ooibc/courses/sql/plsql.htm
- Oracle Database Online Documentation 10g Release 1
  http://docs.oracle.com/cd/B13789_01/index.htm

Valoració

En quant a la bibliografia...

PL/SQL és un llenguatge popular usat en l'entorn empresarial. Com a tal, no falten recursos en línia, llibres ni exemples de codi per aprendre a usar-lo. A la mateixa biblioteca de la FIB n'hi havia en gran quantitat.

Ara bé, he trobat difícil trobar informació exhaustiva sobre funcionalitats concretes. Per posar un exemple, hi ha molta informació sobre les "*bind variables*" i com usar-les, però cada font les mostrava de forma diferent i cap donava una definició clara de que són (és més, un lloc web mencionava que només poden ser de certs tipus de variable, mentre la resta no en deien res). Un altre exemple són els tipus de dades, que són nombrosos i ni tan sols a la documentació d'Oracle estaven tots explicats en detall. Suposo que ho deu motivar el fet que sigui un llenguatge propietari.

En quant al llenguatge...

PL/SQL és un llenguatge fàcil de començar a utilitzar si ja es té experiència amb diversos llenguatges de programació. Ara bé, si realment es vol escriure programes que treguin el màxim rendiment de la base de dades cal experiència per arribar a coneixer-ne tots els trucs.